EVOLUTIONARY MULTI-AGENT TEAMS FOR ADAPTIVE OPTIMIZATION

L. HANNA, Carnegie Mellon University, Pittsburgh, PA.
J. CAGAN,* Carnegie Mellon University, Pittsburgh, PA

ABSTRACT

This paper explores the ability of a team of autonomous software agents to deal with changing optimization environments by evolving to use the most successful algorithms at the points in the optimization process where they will be the most effective. The communal agent team organizational structure employed in this work allows cooperation of agents through the products of their work and creates an ever changing set of individual solutions. An evolutionary approach is used, but evolution occurs at the strategic rather than solution level. As an application of this work, individual solutions will be tours in the familiar combinatorial optimization problem of the traveling salesman. With a constantly changing set of these tours, the team, each agent running a different algorithm, must evolve to apply the solution strategies which are most useful given the set at any point in the process. As a team, the evolutionary agents produced better solutions than any individual algorithm used.

Keywords: Evolutionary agents, adaptive optimization, traveling salesman problem

INTRODUCTION

For many complex optimization problems such as combinatorial optimization problems, exact algorithms and solution strategies for determining the optimal solution often don't exist or are so involved that they are only practical for specific applications under specific conditions. In other words, it is very difficult to determine for each possible starting point in a highly multi-modal design space, what is the best strategy for moving the solution closer to the global optimum. The conditions that motivate using specific solution strategies, if they're even known, may change rapidly as the design space is traversed.

Thus we argue that solution strategies should *evolve* dynamically as conditions change, i.e., as new solution states are discovered during the optimization process, the best strategies may be employed at the correct time to achieve maximum improvement of individual solutions. Evolution is not a new concept, but the use of evolutionary processes on the solution *strategies* is very different from typical genetic algorithms where genetic operators reproduction, mutation, and selection are usually applied to the *solutions*. Here, the solution strategies are recombined, altered, and removed through these genetic operators based on their success in improving solutions.

However, in order to ensure that a globally superior solution is obtained, evolving strategies should also be organized and coordinated in such a way that the design space is explored in as many promising directions as possible when new solutions are presented. The

* Corresponding author address: Jonathan Cagan, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA. 15213; e-mail: cagan@cmu.edu.

idea of cooperation of strategies for design space exploration, in addition to their evolution for maximum effectiveness, has led to the assertion that strategies should be embodied in independent, autonomous software agents, which evolve at a population level to determine the best solution strategy for a given set of solutions but also cooperate to more thoroughly explore the design space.

The evolution of agents, representing solution strategies, at a population level is a rather unique concept. Grefenstette (1992) explored the evolution of solution strategies for predator-prey scenarios, but with the goal of producing a single 'super strategy' from the evolution of a population of strategies (a strategy consisting of a set of decision rules) which could then be applied to the predator-prey scenario (which was the simulated to determine the fitness of individual strategies). Our aim is to evolve an efficient *team* of agents. Because of the constantly changing set of solutions, the presence of agents in the team which run inferior strategies, even in diminished numbers, strengthens the performance of the entire team: again, no one agent can accomplish what the team as a whole is capable of.

TRAVELING SALESMAN PROBLEM

The Traveling Salesman Problem (TSP) was chosen as an application for the proposed framework because it is such a well known and straightforwardly defined problem, though the goal of this work is not to present an algorithm which solves the TSP better than any other algorithm thus far. The objective of the TSP is, given a set of cities and a cost function for each pair of cities, to find the round trip tour with the lowest cost that visits each city once and only once. For the problems we will explore in this paper, the cost, or distance, function between cities is a 'pseudo-euclidean' function described by Padberg and Rinaldi (1987).

Though more successful algorithms have been developed (the reader is referred to Applegate et. al (2006) and Laporte (1992) for descriptions of the best known and most current algorithms), for this work only a few have been chosen in three categories of algorithms, *construction*, *improvement*, and *reduction*. Construction algorithms are so named because they take, as input, an incomplete or partial tour and return either a complete tour or a longer partial tour after adding cities in a predefined manner. For this study three simple and straightforward construction algorithms, nearest insertion, farthest insertion, and arbitrary insertion, were used (Golden et.al. 1985). Improvement algorithms, as their name would suggest, improve an existing partial or complete tour by rearranging the order of the cities in the tour based on different rules. There were three improvement algorithms used in this study, 2-Opt (Bentley, 1990), 3-Opt (Syslo et. al 1983), and a simple mutation. Reduction algorithms break down complete tours into partial tours. In this work, two very basic reduction algorithms are employed. The first of these is random reduction, which involves simply randomly removing a random number of cities in the tour. Best partial reduction, the second reduction algorithm, returns the best partial tour (the tour with the shortest average leg length) containing half of the total number of cities in the same consecutive order as the original tour.

METHODOLOGY

Evolutionary Agents

To perform genetic operations such as crossover and mutation, individuals in an evolving population are most easily represented by binary strings. The binary string defining an individual agent in the evolving team of the proposed framework represents the decisions the agent will make in its lifetime. We argue that decisions should be the primary element of an agent's genetic makeup because autonomy, the ability of an agent to make decisions on its own without being told what to do, is essential to the definition of an agent (Wooldridge et. al. 1995; Sachdev 1998). For the particular application of the TSP, agent decisions were defined as follows:

1. From what memory will a tour be chosen,
2. Which tour from that memory will be worked on,
3. How will the chosen tour be worked on (i.e. which algorithm will be run), and
4. Where (which memory) will the new tour be put once work is completed on it.

Thus, the genetic string of each agent consists of four binary chromosomes identifying these properties (see Figure 1). The choice methods define the characteristics of a tour which an agent will choose, i.e. if the agent will choose the best tour, the worst tour, be biased towards better tours, or be biased towards worse tours. The significance of the memories will be discussed in the next section.

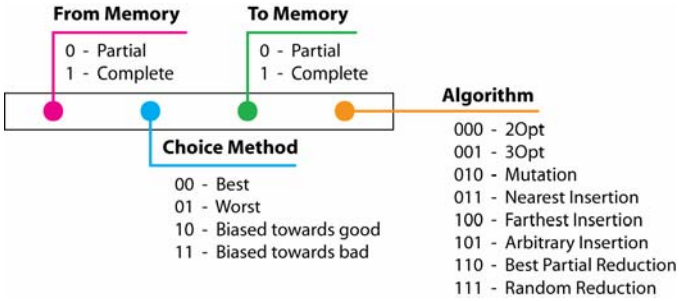

Figure 1 Structure of proposed evolving agent genetic string

Agent Organization: Creating an Evolutionary Multi-Agent Team

The agent system architecture developed is similar to the asynchronous team architecture developed by Talukdar, et. al. (1998) in that it incorporates the idea of shared memories, which allow agents to cooperate indirectly by providing a place for agents to present their work so that it is visible and available to others. However, in those systems the characteristics of each agent and the rules for their relationships to the memories are specified *a priori* (Sachdev 1998; De Souza 1993). In the proposed system the agent-memory cycles are evolved by including input space and output space decisions in the agents' chromosomal representations (toMemory and fromMemory). For the specific application of the Traveling Salesman Problem, only two memories were used: one for partial tours (tours that do not contain all of the cities) and one for complete tours. The tours in these memories evolve over time through the genetically determined actions of the agents, rather than through recombination and mutation within the

population of solutions as would occur in a typical genetic algorithm (Grefenstette et. al. 1985; Potvin 1996).

ALGORITHM DESCRIPTION

Our proposed Evolutionary Multi-Agent System (EMAS) algorithm simulates temporal asynchrony by dividing the overall process into discrete iterations. Each iteration, all agents undergo *activation*, at which point they make decisions and perform actions based on their genetic sequence. After all agents have been activated, *reproduction* occurs, in which parents are selected and new agents are created. Reproduction and activation both involve a simple operator for *mutation*. Finally, the agent community undergoes *selection*, where the weakest individuals are removed from the population. In this section, each of these important functions is discussed in detail.

Mutation

In the proposed framework, mutation is used for two purposes. The first purpose, common to most evolutionary and genetic algorithms, is to make the system more stochastic – mutation allows a more thorough exploration of the design space for individuals by introducing randomness into their creation. In the proposed framework individuals are also mutated when they are not being successful. This secondary mutation is a way of allowing individual agents to adapt to an environment by trying new decision methods, achieving diversity by variation. Both types of mutation are random, meaning that a single randomly chosen bit is altered in the binary gene.

Activation

Each iteration, all agents are activated. Activation of an agent consists of verification that it is able to work (some memory-algorithm combinations are incompatible, i.e. construction cannot be performed on a complete tour) and simulation and testing to determine if it will make a positive difference. Simulation is an important step: agents will not place a solution they know will decrease the average solution quality into their destination memory. This keeps the quality of solutions in the memory high (i.e. keeps the average tour length low). As stated earlier, if an agent is unsuccessful, i.e. unable to improve the average solution quality, after three tries, the agent undergoes mutation. If an agent is successful in coming up with a solution that increases the average solution quality, it then inserts the new tour into its destination memory. A flowchart of agent activation is shown in Figure 2.

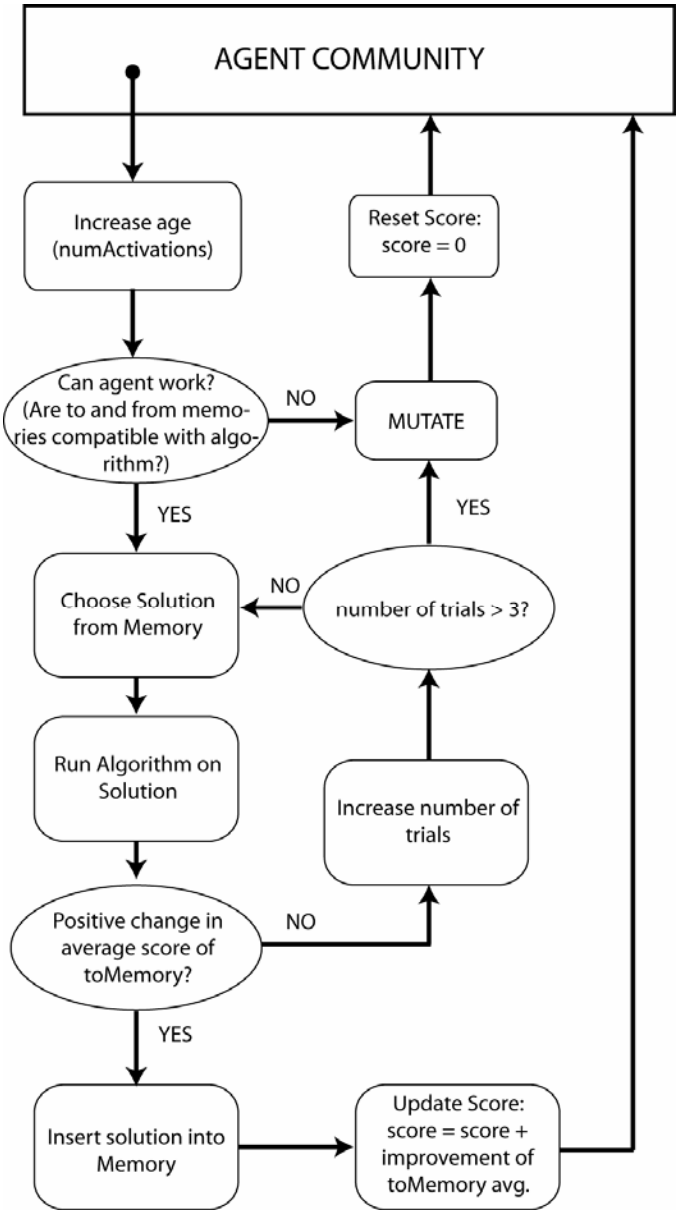

Figure 2 Flowchart of agent activation

Fitness

A key principle in both selection and reproduction is the concept of fitness. It is often difficult to establish a meaningful method for deciding who should live and reproduce and who should not. Thus, before going into detail on the procedure for reproduction and selection, it is important to establish the method of evaluation of individuals. In the proposed framework, the indication of an agent's success is embodied in its score. Score is based both on the amount of improvement made by the agent to the average solution quality in its destination memory and the number of times it has been activated (its 'age'). When agents mutate, their score is reset to zero but their age remains the same.

Reproduction

After activating each agent in an iteration, agents with a score above zero are paired up as parents and allowed to reproduce. Each agent may only reproduce once in an iteration, and during reproduction is subjected to crossover with a randomly assigned partner at a single random crossover point. The resulting two children each have a 50% chance of being mutated. After they are created, the children agents are activated.

Selection

When new agents are added through reproduction, the worst agents are selected from the population to be eliminated, keeping the population size constant. Selection begins by sorting the agents by score from lowest to highest. Agents with the same score are then sorted by age, the oldest on the bottom and the youngest on the top. Once sorting is complete, agents are removed from the bottom of the list until the population is back to its original size.

RESULTS

Though our primary goal in this work was not to develop a method for solving TSP to optimality, the quality of the solutions reached by the evolutionary team of agents proposed in this work was very good compared to the performance of the individual algorithms on their own. The solution quality reached by our Evolutionary Multi-Agent System (EMAS) algorithm were consistently better than those reached by the other base algorithms and hybrid algorithms (*a priori* designated construction algorithm followed by improvement algorithm).

The base construction algorithms nearest and farthest insertion always produce the same final optimization solution for a given starting city, so running these algorithms for each of the starting cities is a good measure of the average effectiveness of each of these base algorithms. Similarly, the same starting tour will always lead to the same final optimization solution after running any of the base improvement algorithms presented. Though the random order of city addition in arbitrary insertion makes the final tour different even for the same starting city, testing each starting city still provides a good estimation for the effectiveness of this algorithm as well. Thus, for the 48-city problem, EMAS was run 48 times (100 iterations each time) and compared to the solutions resulting from running construction algorithms from each starting city and then running improvement algorithms on the resultant tours. Table 1 clearly indicates that, on average, the solution quality produced by the EMAS algorithm is much better than any of these hybrid algorithms. Similarly, the histogram in Figure 3 shows that the majority of solutions reached by the EMAS algorithms were within 1% of optimal, whereas only two of the other hybrid algorithms had any solutions at all in that range.

Table 1 Mean and standard deviation of hybrid algorithms compared to EMAS algorithm for 48-city problem

Algorithm	Mean (% from Optimal)	St. Dev. (%)
3-Opt+NI	3.27	2.67
3-Opt+FI	3.47	1.52
3-Opt+AI	3.08	1.32
2-Opt+NI	9.61	2.32
2-Opt+FI	6.68	1.1
2-Opt+AI	6.09	2.06
EMAS	0.68	0.61

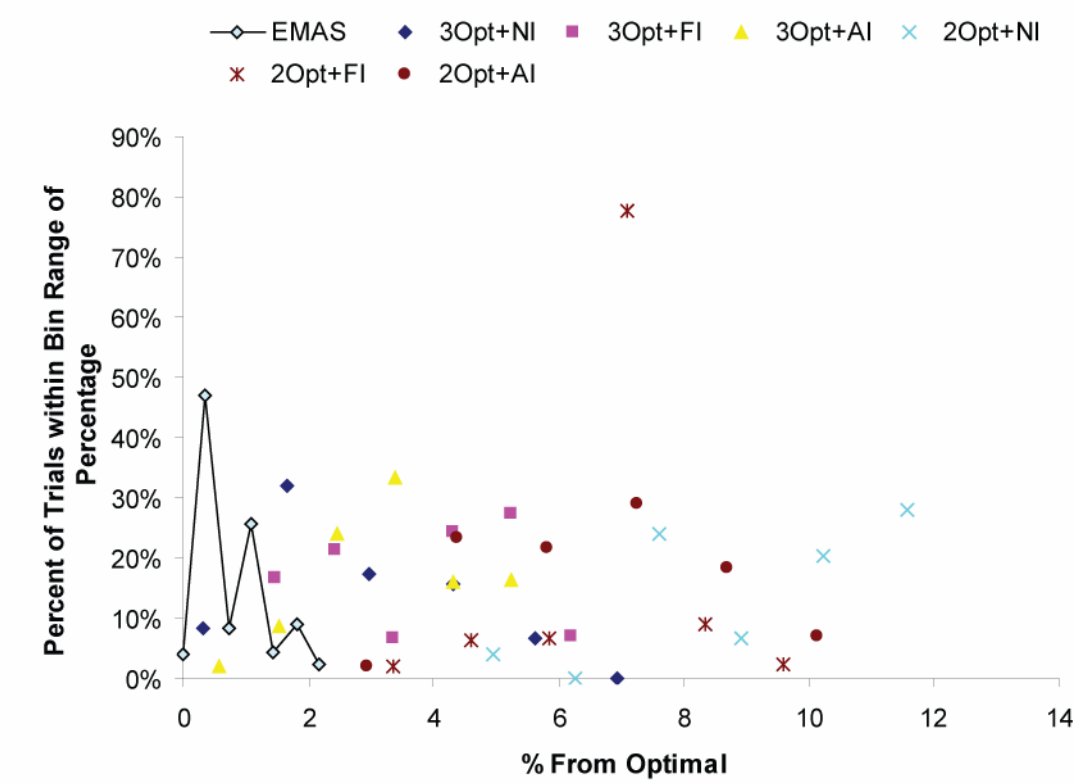

Figure 3 Histogram of 48-city problem comparing best solution consistency of EMAS to that of hybrid algorithms

The consequence of this increased quality of solutions was computation time. Because EMAS involves running several of the base algorithms each iteration, it is expected that the amount of time required to reach the solutions generated is much higher. A single run of 3-Opt on any individual starting tour for ATT48 would take less than a second, whereas a single trial of EMAS run on ATT48 for 100 iterations takes an average of around 8 seconds. As mentioned earlier, however, it doesn't matter how many times this algorithm is run on the same starting

tour, it will always produce the same final tour, which as we have just shown for the base algorithms (construction only and improvement only) is usually worse than the result of the EMAS algorithm. We show in Table 3, however, that even running the same number of algorithms as would be run during a single trial of EMAS (10 algorithms * 100 iterations) in random order without employing the evolutionary aspect of the EMAS algorithm will still result in worse solutions.

Table 2 Comparison of EMAS to randomly generated algorithm activation order for ATT48 (Averages of 50 trials)

	Mean (% from Optimal of average tour in Complete memory)	St. Dev. (%)	Avg. Time (sec)
1000 Randomly Ordered Algorithms	4.83	0.354	2.45
EMAS	4.38	1.66	8.33

SUMMARY AND DISCUSSION

The results present a convincing argument for the evolution of agents in a team at the population level. Decisions have likewise proven to be a useful genetic property of agents in such an evolutionary setting. The evolutionary teams evolved to generate better solutions than the base algorithms alone. We have also shown that the strength of the EMAS algorithm lies in its ability to evolve the best team each iteration. Evolution and activation within this team results in solutions that are better than simply running the same number of algorithms randomly on a similar set of solutions. We thus argue that the use of evolutionary agents to determine the best solution strategies dynamically is a strong approach to adaptive optimization.

We have also begun to test this strategy with a much larger, 532 city TSP with even better results in terms of solution quality. In so doing we have identified patterns in how the EMAS algorithm allocates types of agents throughout its run. We hypothesize that we can take advantage of such patterns to improve run time in future work.

Acknowledgements

This research was sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA95500710225. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR or the U.S. Government.

REFERENCES

Applegate, D.L., R.E. Bixby, V. Chvatal, and Cook, W.J. 2006. *The Traveling Salesman Problem*. Princeton University Press, New Jersey, USA.

Bentley, J.L. 1990. "Experiments on Traveling Salesman Heuristics," *Proc. 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, Philadelphia, PA. pp. 91-99.

De Souza, P.S. 1993. "Asynchronous Organizations for Multi-Algorithm Problems" Ph.D. Dissertation, Carnegie Mellon University, Department of Electrical and Computer Engineering.

Golden, B.L. and W.R. Stewart, 1985. "Empirical Analysis of Heuristics," in *The Traveling Salesman Problem*, E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, and D.B. Shmoys, eds. John Wiley.

Grefenstette, J.J., R. Gopal, B.J. Rosmaita, and D. Van Gucht. 1985. "Genetic Algorithms for the Traveling Salesman Problem". *Proc. 1st Int'l Conf. on Genetic Algorithms*. pp. 160-168.

Grefenstette, J.J. 1992. "The Evolution of Strategies for Multi-Agent Environments". *Adaptive Behavior* **1** (1) pp. 65-89.

Laporte, G. 1992. "Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms". *European Journal of Operational Research*. Vol. 59, no. 2, pp. 231-247.

Padberg, M. and G. Rinaldi. 1987. "Optimization of a 532-city symmetric traveling salesman problem by branch and cut." *Operations Research Letters* **6**, 1-7.

Potvin, J. 1996. "Genetic Algorithms for the Traveling Salesman Problem". *Annals of Operations Research*, Vol. 63, no. 3. pp. 337-370.

Sachdev, S. 1998. "Explorations in Asynchronous Teams" Ph.D. Dissertation, Carnegie Mellon University, Department of Electrical and Computer Engineering.

Syslo, M.M., N. Deo, and J.S. Kowalik, 1983. "Discrete Optimization Algorithms with Pascal Programs," Prentice Hall, Englewood Cliffs, NJ.

Talukdar, S., L. Baerentzen, A. Gove, P. De Souza. 1998. "Asynchronous Teams: Cooperation Schemes for Autonomous Agents". *Journal of Heuristics*, **4**. 295-321.

Wooldridge, M. and N.R. Jennings. 1995. "Intelligent Agents: Theory and Practice" *The Knowledge Engineering Review* **10** (2), 115-152.